SPECIFICATION AMENDMENTS

[0016] In ~~VM's~~ VMs that support a plurality of I/O modes (such as audio and video, or from a network to video, etc.) the invention may filter the I/O data with respect to a criterion relating to one of the modes but transform data expressed in a form corresponding to a different one of the I/O modes.

[0028] Some interface is usually required between a VM and the underlying "real" OS 220 (real in the sense of being either the native OS of the underlying physical computer, or the OS or other system-level software that handles actual I/O operations, takes faults and interrupts, etc.) and hardware, which are responsible for actually executing VM-issued instructions and transferring data to and from the actual, physical memory and storage devices 112, 114. This interface is often referred to as a virtual machine monitor (VMM). A VMM is usually a thin piece of software that runs directly on top of a host, or directly on the hardware, and virtualizes all, or at least some subset of, the resources of the machine. The interface exported to the respective VM is the same as the hardware interface of the machine, or at least of *some* predefined hardware platform, so that the virtual OS cannot determine the presence of the VMM. The VMM also usually tracks and either forwards (to the OS 220) or itself schedules and handles all requests by its VM for machine resources as well as various faults and interrupts. The general features of ~~VMM's~~ VMMs are known in the art and are therefore not discussed in detail here.

[0029] In Figure 1, ~~VMM's~~ VMMs 400, ..., 400n, are shown, acting as interfaces for their respective attached ~~VM's~~ VMs 300, ..., 300n. It would also be possible to include each VMM as part of its respective VM, that is, in each virtual system. Moreover, it would also be possible to use a single VMM to act as the interface to all ~~VM's~~ VMs, although it will in many cases be more difficult to switch between the different contexts of the various ~~VM's~~ VMs (for example, if different ~~VM's~~ VMs use different virtual operating systems) than it is simply to include a separate VMM for each VM. The important point is simply that some well-defined, known interface should be provided

between each virtual system 300, ..., 300n and the underlying system hardware 100 and software 220.

[0030] In some conventional systems, ~~VMM's~~ VMMs will run directly on the underlying system hardware 100, and will thus act as the "real" operating system for its associated VM. In other systems, the OS 220 is interposed as a software layer between ~~VMM's~~ VMMs and the hardware; still other arrangements are possible. This invention works with all such configurations, the only requirement being that I/O requests by any VM, and the returned results of the requests, should be able to be tracked and intercepted by some software component that performs the functions of the VMM described below.

[0092] The transformation implemented by the invention need not involve any form of visible display. Encryption/decryption is one example of this. As one other example, the transformation module 474 may be programmed, using conventional techniques, to act as a network bandwidth limiter or network traffic shaper, for example, in order to allocate network bandwidth to different ~~VM's~~ VMs. For example, each VMM could limit its respective VM to a predetermined maximum bandwidth such as 1MB/s; all I/O requests requiring more than 1MB/s of bandwidth could then be limited simply by having the transformation module 474 drop packets that exceed the 1MB/s bandwidth limit. Different ~~VMM's~~ VMMs could also cooperate in such as way as to dynamically adjust the bandwidth allocation for different ~~VM's~~ VMs as a function of current availability.